Reliability/Redundancy Trade-off Evaluation for Multiplexed Architectures used to Implement Quantum Dot Based Computing

(Design Tools Track: Design Validation, Fault Tolerance)

Abstract

With the advent of nanocomputing, researchers have proposed Quantum Dot Cellular Automata (QCA) as one of the implementation technologies. The majority gate is one of the fundamental gates implementable with QCAs. Moreover, majority gates play an important role in defect- and fault-tolerant circuit implementations for nanotechnologies due to their use in redundancy mechanisms such as TMR, CTMR etc. Therefore, providing reliable implementation of majority logic using some redundancy mechanism is extremely important. This problem was addressed by von Neumann in 1956, in the form of ”majority multiplexing” and since then several analytical probabilistic models have been proposed to analyze majority multiplexing circuits. However, such analytical approaches are extremely challenging combinatorially and error prone. Also the previous analyses did not distinguish between permanent faults at the gates and transient faults due to noisy interconnects or noise effects on gates. In this paper, we provide explicit fault models for transient and permanent errors at the gates and noise effects at the interconnects. We model majority multiplexing in a probabilistic system description language, and use probabilistic model checking to analyze the effects of our fault models on the different reliability/redundancy trade-offs for majority multiplexing configurations. We also draw parallels with another fundamental logic gate multiplexing technique, namely NAND multiplexing. Tools and methodologies for analyzing redundant architectures that use majority gates will help logic designers to quickly evaluate the amount of redundancy needed to achieve a given level of reliability. VLSI designs at the nano-scale will utilize implementation fabrics prone to faults of permanent and transient nature, and the interconnects will be extensively affected by noise, hence the need for tools that can capture probabilistically quantified fault models and provide quick evaluation of the trade-offs. A comparative study of NAND multiplexing vs. majority multiplexing is also needed in case the designers are confronted with the choice to implement redundancy in either way. This paper provides models, methodologies and tools for these much needed analyses.[1]

Index Terms

Probabilistic model checking, probability, nanotechnology, multiplexing, reliability, majority, defect-tolerance, fault-tolerance, transient errors, interconnect noise

I. Introduction

Creating computing systems built from devices other than CMOS transistors is a research area of major interest. A number of alternatives to silicon VLSI have been proposed, including techniques based on molecular electronics, quantum mechanics, and biological processes. One such transistor-less nanostructure proposed by Lent et al. [1], [2] is the quantum-dot cellular automata (QCA) that employs arrays of electrostatically coupled quantum dots to implement Boolean networks. Due to the minuscule size of the quantum dots, extremely high packing densities are possible for QCA based architectures. Also, the other advantages are simplified interconnections due to near-neighbor interactions, extremely low power-delay product and decreased heat dissipation. The fundamental QCA logic device is a three-input majority logic gate shown in Figure 1 (a) consisting of an arrangement of

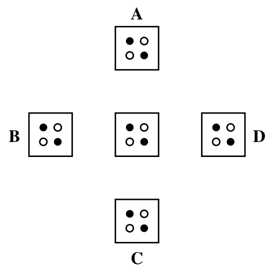
[1] **LA-UR-04-4882**

five standard quantum cells: a central logic cell, three inputs (A, B and C) and an output cell. A majority gate can be programmed to act as an OR gate or an AND gate, by using one of the three inputs as the control input that determines the AND/OR functionality [3]. This indicates the importance of majority gates in QCA based architectural configurations. Majority gates are also used in different conventional resource redundancy based techniques such as TMR, CTMR, multi-stage iterations of these etc. In computation fabrics based on nanoscale devices, the probability of the devices and interconnects being faulty (manufacturing or transient) will be non-negligible; in fact, faults may be common. This implies that hardware redundancy based techniques (TMR/CTMR) will be fundamental in designing fault- and defect-tolerant nanoarchitectures—hence the significance of majority gates. Since majority gates will be heavily used in QCA-based or any other redundant, nanoscale architectures, tools and techniques are needed to model probabilistically quantified fault models and calculate reliability-redundancy trade-off points for specific architectures. This motivates our work of automating the detailed reliability analysis of the von Neumann majority multiplexing architecture.

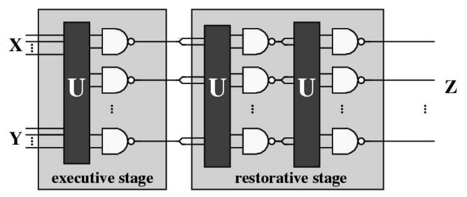
In 1952, von Neumann introduced a resource redundancy based technique called *multiplexing* [4] for constructing reliable computation from unreliable devices. He considered two sets of basic logic in his architectural configurations, the NAND (Figure 1 (b)) and the majority logic (Figure 2). He showed that such multiplexing-based architectures can perform computations with high reliability, if the failure probabilities of the gates are sufficiently small and failures are independent. Pippenger [5] showed that von Neumann's construction works only when the probability of failure per gate is strictly less than $1/2$, and that computation in the presence of noise (which can be seen as the presence of transient fault) requires more layers of redundancy. [6] also showed that a logarithmic redundancy is necessary for some Boolean function computations and is sufficient for all Boolean functions. In-depth analysis of the NAND multiplexing technique in [7] shows that each specific architecture has *reliability-redundancy* trade-off points and that the arbitrary augmentation of unreliable devices could result in degradation of the reliability measures of an architecture. This means that for given failure distribution of devices, once an optimal redundancy level is reached, any increase or decrease in the number of devices may lead to less reliable computation. It has also been shown in [8] that redundancy may also be applied at different levels of granularity, and there exists redundancy vs. reliability vs. granularity trade-off points. Determining such trade-off points expedites design and evaluation of different defect- and fault-tolerant architectures.

Theoretical models have also been proposed in the past [9], [10], and results from information theory [5], [11] have been used to analyze redundancy-reliability trade-offs of fault-tolerant architectures. Analytical techniques to compute the reliability measures of considerably large architectures built with majority gates would require very involved combinatorial arguments. Also, such analytical probabilistic analysis for large multiplexed configurations is often non-composable in the sense that if the analyzed configuration is used as a part of a larger configuration, the combinatorial analysis becomes much more difficult. Also, if there are interdependencies between the analyzed multiplexed gate and parts of of a larger multiplexing configuration, the probabilistic analysis has to be done from the beginning. As a result, the analysis in [12] is difficult to use when a large number of majority gates/multiplexing stages are used in a larger design. As another example of the complexity of direct analysis, it has been observed that the authors of [9] made a mistake in calculating the reliability of NAND multiplexing structures since they applied selection with replacement in their combinatorial argument where the appropriate one would be without replacement. This kind of error can be avoided if the system being analyzed can be modeled with interacting probabilistic transition systems, and the tool can analyze the probabilities automatically by constructing the corresponding Markov chains/processes and applying state space traversal techniques. By using guarded commands in our framework description language, the models are composable, and corresponding Markov analysis is based on the ability of probabilistic model checkers to handle large state spaces (which continually improves [13]). Our approach entails such tools and techniques that help in avoiding combinatorial arguments, which is exactly where the merit of this approach lies. The effect on the reliability of a system due to variations in the behavior of the system's components—for example, the change in reliability as the probability of gate failure or interconnect noise varies—can also be easily computed by such a methodology.

In this paper, we analyze reliability measures of multiplexing based majority circuits when these are subjected to specific fault models. To achieve different reliability/redundancy trade-off figures and compare these against previously reported figures for NAND-multiplexing [7], we have enhanced our probabilistic model checking based tool NANOPRISM [8] by building a generic multiplexing library. This library can be used used to add multiplexing based hardware redundancy in arbitrary Boolean networks, and at different levels of granularity such as the gate, logic block, logic function, and unit levels. Also, explicit fault models have been added to our tool NANOPRISM. These fault models are failure of logic gates to compute the correct Boolean output (assume a probability of gate failure, covering both transient or permanent failure with different probability values) and interconnect failure (noise or other perturbations that may change the probability of a signal being at logic low or high at the interconnects between different stages). The formulation of such probabilistically quantified fault models allow us to distinguish between permanent and transient faults. The multiplexing library computes probability distributions at the ouputs of the different multiplexing architectures and these are used in analyzing reliability/redundancy trade-off points.



(a) QCA majority gate configuration



(b) A NAND multiplexing unit

Fig. 1. QCA majority gate and NAND multipexing unit

A. Main Results

We have enhanced our reliability analysis tool NANOPRISM (described in brief later). The enhancements are as follows:

- a generic multiplexing library has been added, such that this resource redundancy technique can be used in a Boolean network, or in a selected portion of a logic network.
- explicit gate and interconnect fault models have been added and different probability distributions may be used to represent these faults.
- permanent and transient faults can be injected in the design, either separately or collectively.

We have also determined different reliability measures for multiplexing architectures based on majority circuits. Also the results of different configurations for majority and NAND multiplexing have been compared. Our observations show that:

- for large gate failure probabilities, either the reliability of the system does not improve or may even *degrade* when adding more redundant devices, (this can be observed in Figure 6 for gate failure probability of 0.04);
- for the same redundancy factors, the majority multiplexing system is less reliable than NAND multiplexing when the probability of the gates failing is small (Figure 9);
- majority multiplexing provides better reliability when compared to NAND multiplexing systems for large gate failure probabilities (Figure 10); and,
- in the presence of interconnect noise, increasing the number of majority gates in each multiplexing stage results in decrease in reliability, whereas, increasing the number of multiplexing stages improves reliability measures(Figure 8 and Figure 7 (b)).

Such observations could be of significant consequence in determining redundancy levels for systems with majority gates as the basic building blocks.

B. Organization

The organization of the paper is as follows: the next section introduces the basic concepts of fault-tolerant computing, von Neumann multiplexing, probabilistic model checking and the NANOPRISM tool. In Section 3 we describe how we use the PRISM framework to model the majority multiplexing fault-tolerant architecture. Section 4 reports on the experimental results obtained for the different case studies. We compare reliability measures for the NAND and majority-gate multiplexing architectures in Section 5 and, finally, Section 6 summarizes the results of this work.

II. Background

In this section, we briefly outline basic concepts from fault-tolerant computing, von Neumann multiplexing, probabilistic model checking, the probabilistic model checker *PRISM*, and our tool NANOPRISM.

A. Defect-Tolerant Computing

Formally, a *fault-tolerant architecture* is one which uses techniques to mitigate the effects of faults in the devices that make up the architecture and guarantees a given level of reliability. There are a number of canonical redundancy-based fault-tolerant architectures in the fault-tolerant literature. We discuss Triple Modular Redundancy, Cascaded Triple Modular Redundancy and multi-stage iterations of these. In the following subsection, we discuss the details of the von Neumann multiplexing scheme.

The concept of Triple Modular Redundancy (TMR) is to have three units working in parallel, comparing their outputs with majority voting logic. TMR provides a functionality similar to one of the three parallel units but provides a better probability of working. The trade-off is that instead of n devices, 3n devices and a majority gate are needed to make this configuration. The R-fold modular redundancy is a generalization of the TMR configuration where R units work in parallel instead of 3 and R $\in \{3, 5, 7, 9.....\}$.

The concept of Cascaded Triple Modular Redundancy (CTMR) is similar to TMR, wherein the units working in parallel are TMR units combined with a majority gate. This configuration forms a CTMR of the first order and therefore TMR can be considered to be 0th order CTMR. Higher orders of CTMR are obtained by multi-stage iterations of these, but this does not necessarily mean that the reliability of a system improves. This is because increasing the redundancy level also introduces more unreliable devices in the architectural configuration.

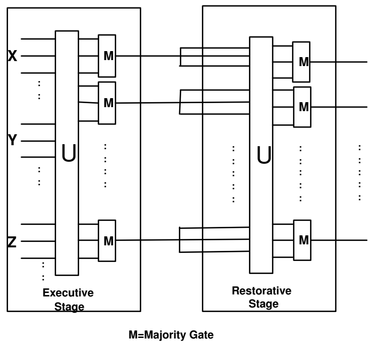


Fig. 2. A majority multiplexing unit

B. Multiplexing Based Fault-Tolerance

The multiplexing based fault-tolerance scheme replaces a single logic device by a multiplexed unit with N copies of every input and output of the device. The N devices in the multiplexing unit process the copies of the inputs in parallel to give N outputs. Each element of the output set will be identical and equal to the original output of the logic device, if all the copies of the inputs and devices are reliable. However, if the inputs or devices are in error, the outputs will not be identical. To tackle such an error-prone scenario, some critical level $\Delta \in (0, 0.5)$ is defined. The output of the multiplexing unit is considered stimulated (taking logical value `true`) if at least $(1-\Delta)\cdot N$ of the outputs are stimulated and non-stimulated (taking logical value `false`) if no more than $\Delta \cdot N$ outputs are stimulated.

If the number of stimulated outputs is in the interval $(\Delta{\cdot}N, (1{-}\Delta){\cdot}N)$, then the output is undecided, and hence a malfunction will occur. The basic design of a von Neumann multiplexing technique consists of two stages: the *executive stage*, which performs the basic function of the processing unit to be replaced, and the *restorative stage*, which reduces the degradation in the executive stage caused by erroneous inputs and faulty devices. Figure 1 (b) shows the architectural configuration when a NAND gate is the processing unit of a multiplexing system.

In this paper, we consider multiplexing when the logic device is a single majority gate. We therefore replace the inputs and output of the gate with N copies and duplicate the majority gate N times in the executive stage, as shown in Figure 2. The unit U represents a *random permutation* of the input signals, that is, for each input set to the N copies of the majority gate, three input signals are randomly chosen from the three separate input bundles respectively. Figure 2 also shows the restorative stage which is made using the same technique as the executive stage, duplicating the outputs of the executive stage to use as its inputs. Note that applying this approach only once will invert the result when using NANDs, therefore two steps are required. To give a more effective restoration mechanism this stage can be iterated [4].

C. Probabilistic Model Checking and PRISM

Probabilistic model checking is a range of techniques for calculating the likelihood of the occurrence of certain events during the execution of unreliable or unpredictable systems. The system is usually specified as a state transition system, with probability values attached to the transitions. A probabilistic model checker applies algorithmic techniques to analyze the state space and calculate performance measures. We use PRISM [13], [14], a probabilistic model checker developed at the University of Birmingham. It supports analysis of three types of probabilistic models: discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs) and Markov decision processes (MDPs). We use DTMCs to model the generic multiplexing library for majority logic gates. This model of computation specifies the probability of transitions between states, such that the probabilities of performing a transition from any given state sums up to 1. DTMCs are suitable for conventional digital circuits and the fault models considered. The fault models are manufacturing defects in the gates and transient errors that can occur at any point of time in a Boolean network.

The PRISM description language is a high-level language based on guarded commands. The basic components of the language are *modules* and *variables*. A system is constructed as a number of modules which can interact with each other by means of the standard process algebraic operations [15]. A module contains a number of variables which represents its state. Its behavior is given by a set of guarded commands of the form:

```
[] <guard> → <command>;
```

The guard is a predicate over the variables of the system and the command describes a transition which the module can make if the guard is true (using primed variables to denote the next values of variables).

The properties of a DTMC model can be verified by PCTL model checking techniques. PCTL [16] is a probabilistic temporal logic, an extension of CTL. It can be used to express properties such as ”termination will eventually occur with probability at least 0.98”. Our work is based on modeling defect- and fault-tolerant architectures as state machines with probabilistic assumptions about the occurrence of faults in the devices and interconnections, and then use Markov analysis and probabilistic model checking techniques to evaluate reliability measures.

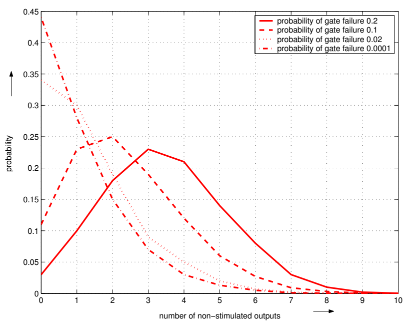
D. NANOPRISM: A Brief Description

NANOPRISM [8] is a probabilistic model checking based tool built on PRISM, that applies probabilistic model checking techniques to compute the reliability/redundancy trade-off points of different Boolean networks that are subjected to probabilistically quantified fault models such as manufacturing and transient faults in devices and interconnections of nano architectures. It consists of libraries for different hardware-redundancy-based, fault-tolerant techniques such as TMR and CTMR. These libraries also support modeling of these redundancy techniques at different levels of granularity, such as the gate, logic block, logic function, and unit levels. Arbitrary Boolean networks are given as inputs to these libraries and reliability measures of these circuits are evaluated at different redundancy and granularity levels. This is the first such tool that automates the evaluation of redundancy vs. reliability vs. granularity trade-offs. Even in [7], there is concentration only on a specific redundancy technique (namely NAND multiplexing), and only redundancy vs. reliability trade-offs are analyzed. All previous work known to us in evaluating reliability measures of systems in the presence of errors inherent in nanotechnology are analytical, and none considered granularity as a parameter. Using this tool, we have illustrated some anomalous, counter-intuitive trade-off points [8] that would not be possible to observe without significant and extensive theoretical analysis. Determining such reliability-redundancy trade-offs and saturation points efficiently and quickly will provide hardware/system designers better insight into fault-tolerant design decisions.

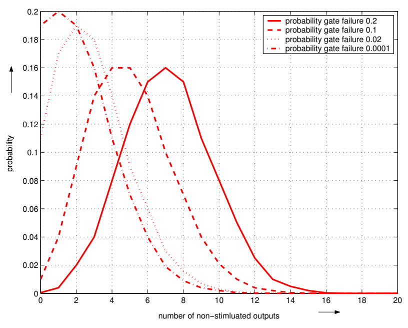
III. Model Construction of Majority Multiplexing System

In this section, we explain the PRISM model of a majority gate multiplexing configuration. The first approach is directly modeling the system. A PRISM module is constructed for each multiplexing stage comprised of N majority gates and these modules are combined through synchronous parallel composition. However, such a model construction scheme leads to the well know state space explosion problem. At the same time, we observe that the actual values of the inputs and outputs for each stage is not important, instead one needs to keep track of only the total number of stimulated (and non-stimulated) inputs and outputs. Furthermore, to allow us to compute these values, without having to store all the outputs of the majority gates in each stage, we replace the set of N majority gates working in *parallel* with N majority gates working in *sequence*. The same methodology is applied to the multiplexing stages of the system so as to reuse the same module for each of the stages while keeping a record of the outputs from the previous stage. This folds space into time by reusing the same majority gate/stage over time rather than making multiple instances of each gate or stage. This approach does not influence the performance of the system since each majority gate works independently and the probability of each gate failing is also independent.

Let us discuss how the unit U indicated in Figure 2 performs random permutation. Consider the case when k outputs from the previous stage are stimulated for some $0 < k < N$. Since there are k stimulated outputs, the next stage will have k of the inputs stimulated if U performs random permutation. Therefore, the probability of either all or none of inputs being stimulated is 0. This implies that each of the majority gates in a stage are dependent on one other, for example, if one majority gate has a stimulated input, then the probability of another having the same input stimulated decreases. It is difficult to calculate the reliability of a system by means of analytical techniques for such a scenario. To change the number of restorative stages, bundle size, the probabilities of the input signals, and the probability of the majority gates or the interconnects failing requires only modification of parameters in the PRISM model description. Since PRISM can also represent non-deterministic behavior, one can set upper and lower bounds on the probability of gate/interconnect failure and then obtain best- and worst-case reliability characteristics for the system under these bounds. The models for the different multiplexing configurations are verified for PCTL [16] properties such as ”probability of having less than 10% errors at the primary output”.

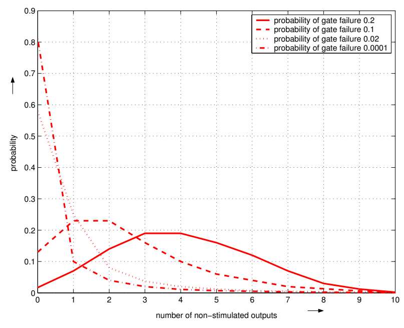


(a) I/O bundle size equals 10

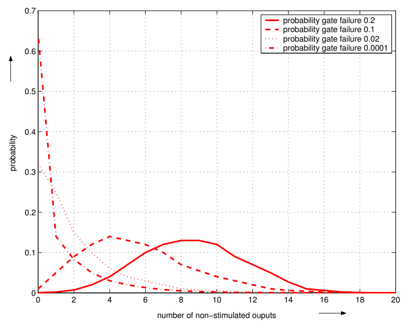


(b) I/O bundle size equals 20

Fig. 3. Error distributions at the output with 1 restorative stage under different gate failure rates and I/O bundle sizes



(a) I/O bundle size equals 10



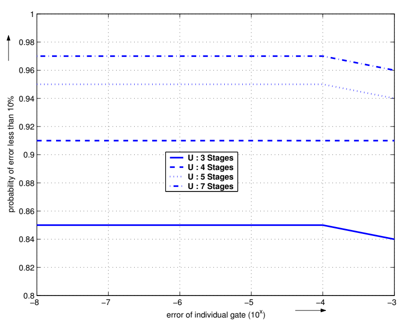
(b) I/O bundle size equals 20

Fig. 4. Error distributions at the output with 3 restorative stages under different gate failure rates and I/O bundle sizes

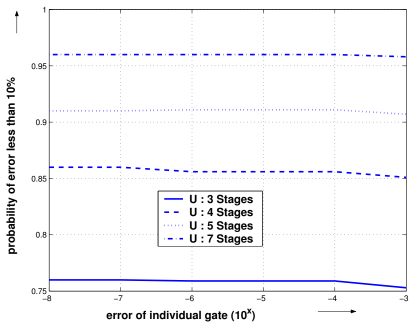
IV. Reliability Evaluation of Multiplexing based Majority Systems

In this section we study the reliability measures of multiplexing based majority systems both when the I/O bundles are of size 10 and 20. These bundle sizes are only for illustration purposes and we have investigated the performance of these systems for larger bundle sizes. In all the experiments reported in this paper, we assume that the inputs X, Y and Z are identical (this is often true in circuits containing similar devices) and that two of the inputs have high probability of being logic high (0.9) while the third input has a 0.9 probability of being a logic low. Thus the circuit's correct output should be stimulated. Also, it is assumed that the gate failure is a von Neumann fault, i.e., when a gate fails, the value of its output is inverted.

PRISM is used to compute the error distribution at the output of the majority gate multiplexing architectures. Hence any measure of reliability can be calculated from these results. PRISM can also be used to directly compute other reliability measures such as, the probability of errors being less than 10% and the expected number of incorrect outputs of the system. Our analysis of reliability for the majority multiplexing system using the NANOPRISM generic multiplexing library concentrates on the effects of the failure probabilities of the majority gates and the number of restorative stages. The results we present show:



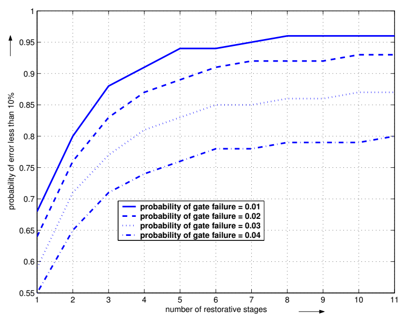
(a) I/O bundle size equals 10



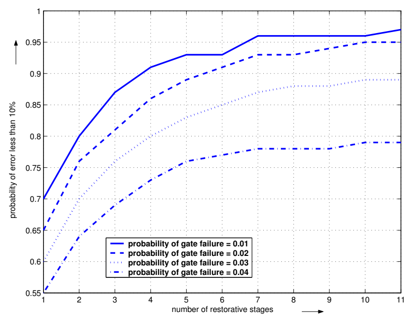
(b) I/O bundle size equals 20

Fig. 5. Probability that at most 10% of the outputs being incorrect for different I/O bundle sizes (small probability of failure)

- the error distribution at the output for different gate failure probabilities (Figure 3) and when more restorative stages are added (Figure 4);
- reliability in terms of the probability that, at most, 10% of the outputs are erroneous for a range of gate failure probabilities (Figure 5);
- reliability almost reaches a steady state for small gate failure probabilities and can be improved marginally once a certain number of restorative stages have been added to the architecture (Figure 5);
- the maximum probability of gate failure allowed for the system to function reliably by comparing the probability that at most 10% of the outputs are incorrect and the expected percentage of incorrect outputs for different numbers of restorative stages (Figures 6 and 7 (a)); and,
- the presence of interconnect noise in a system requires that more layers of redundancy (the number of restorative



(a) I/O bundle size equals 10



(b) I/O bundle size equals 20

Fig. 6. Probability that at most 10% of the outputs of the overall system are incorrect for different I/O bundle sizes (large probability of failure)

stages) be added to maintain the same reliability level as the bundle size is increased.(Figure 8 and Figure 7 (b)).

A. Error Distribution at the Outputs of Different Configurations

In the following analyses, we consider majority multiplexing systems, as shown in Figure 2, where the I/O bundle sizes equal 10 and 20. We first investigate the effect of varying the failure probabilities of the majority gates on the reliability of the system. Figure 3 shows the error distribution at the output of the system in the cases when the probability of gate failure equals 0.2, 0.1, 0.02 and 0.0001. The experimental setup is such that two of the inputs of the majority gate are stimulated when working correctly, and the correct output of the majority gate should be logic high. Hence, when more outputs are non-stimulated, system reliability decreases.

As expected, the distributions given in Figure 3 show that as the probability of a gate failure decreases, the reliability of the multiplexing system increases, i.e., the probability of the system returning incorrect results diminishes. If Figure 3 (a) and (b) are compared, it can be determined that the increase in I/O bundle size increases the reliability of the system as the probability of having erroneous outputs decreases.

When more restorative stages are added to the architecture, a change in reliability is observed. Figure 4 presents the error distributions at the output of the system with 3 restorative stages. The gate failure probabilities are similar to Figure 3. Comparing these distributions with those presented in Figure 3, we observe that, when the gate failure probability is sufficiently small (e.g., 0.0001), adding more restorative stages results in increased reliability, i.e., the probability of non-stimulated outputs is small. On the other hand, in the cases when the gate failure probability is sufficiently large, adding stages does not increase reliability and, in fact, can decrease the reliability of the system (compare the distributions when the failure probability equals 0.2 for each bundle size).

B. Small Gate Failure Probabilities

In Figure 5, we have plotted the probability that less than 10% of the outputs are incorrect against small gate failure probabilities for multiplexing with 3, 4, 5 and 7 stages. These plots show that for small gate failure probabilities, the reliability of the multiplexing system almost reaches a steady state. Comparing Figure 5(a) and (b), it can be inferred that increasing the bundle size results in improvement of the system's reliability. We have also experimented with higher I/O bundle sizes such as 40 and 45, and the results from these multiplexing configurations show that the rate of increase in reliability of the system decreases as the bundle size increases.

Also, these results demonstrate that increasing the number of stages can greatly enhance the reliability of the system. However, the degree of improvement slows down as more restorative stages are added. Moreover, there exists an optimal redundancy level (in terms of the number of restorative stages) beyond which the reliability improvement is marginal. For example, let us consider the plots presented in Figure 5 that indicate probability of erroneous outputs being less than 10% when the number of restorative stages equals 5 and 7. These show that the probability values increase marginally for different gate failure probabilities as compared to when the architecture has a lower number of restorative stages. We should also mention that this result corresponds to the observation made in [9] that, as the number of stages increases, the output distribution of the system will eventually become stable and independent of the number of stages.

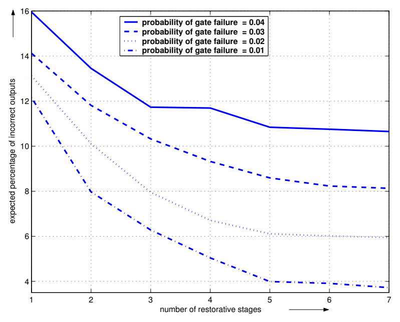
C. Large Gate Failure Probabilities

In this subsection, we consider the case when the probability of gate failure of the majority gates becomes too large for the multiplexing system to function reliably. Figure 6 plots the probability that less than 10% of the outputs are incorrect against large gate failure probabilities (between 0.01 and 0.04) for different numbers of multiplexing stages $\in \{1, 2, \cdots, 10, 11\}$. As can be seen from the results, when the probability of gate failure is equal to 0.04, increasing the I/O bundle size does not improve the reliability of the system. Comparing the same plots in Figure 6 (a) and (b), it can be observed that adding more restorative stages and increasing the I/O bundle size of the architecture tends to cause a degradation in reliability of computation. This anomalous behavior can be understood as follows: when the failure rate is 0.04 (or higher), the restorative stages are sufficiently affected by the probability of gate failures to be unable to reduce the degradation, and hence increasing the number of stages in this case makes the system more error prone.

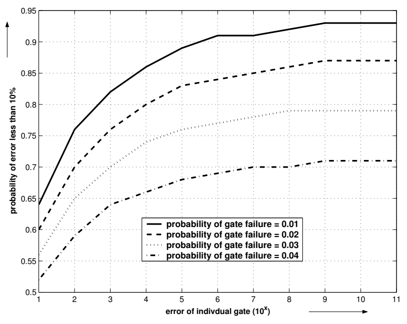
We can infer from these observations that for gate probabilities of 0.04 and higher, increasing bundle size or adding more restorative stages cannot make the system more reliable and may even cause degradation. On the other hand, in the case when the gate failure probability is less than 0.04, the results demonstrate that the system can be made reliable once a sufficient number of restorative stages has been added.

In Figure 7 (a), we have plotted the expected percentage of incorrect outputs when I/O bundle size equals 10. The gate failure probabilities and restorative stages are configured similar to those found in Figure 6. The plots in Figures 7 (a) and 6 suggest similar intuitive results but provide different perspectives to the reliability evaluation of the system. By similar results, we mean that for gate failure probabilities of 0.04 and higher, increasing the number of restorative stages cannot improve reliability of the system to a very high level. In fact, when the multiplexing system is configured to have a specific number of restorative stages, the reliability may decrease as compared to a system with less redundancy (fewer multiplexing stages). It can also be observed from Figure 7 (a) that for all gate failure probabilities, the reliability of the architecture reaches a steady state once a sufficient number of restorative stages have been added.

It is important to note that there is a difference between the bounds computed by our NANOPRISM tool on the probability of gate failure required for reliable computation and the theoretical bounds presented in the literature. This difference is to be expected since our methodology evaluates the performance of systems under fixed configurations, whereas the bounds presented in the literature correspond to scenarios where different redundancy parameters can be increased arbitrarily in order to achieve a reliable system.

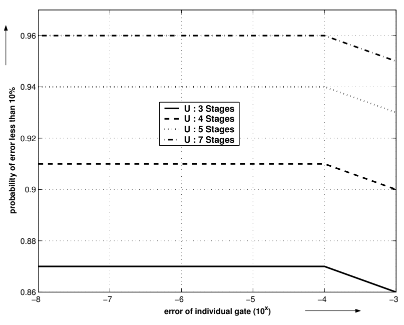


(a) Expected percentage of incorrect outputs

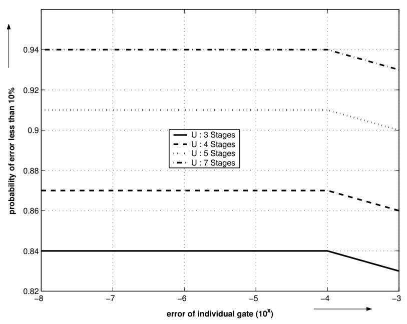


(b) Probability that at most 10% of the outputs are incorrect in the presence of signal noise

Fig. 7. Reliability measures for I/O bundle size of 10 (large probability of failure)



(a) I/O bundle size equals 10



(b) I/O bundle size equals 20

Fig. 8. Probability that at most 10% of the outputs being incorrect in the presence of signal noise (small probability of failure)

D. Modeling of Interconnect Noise

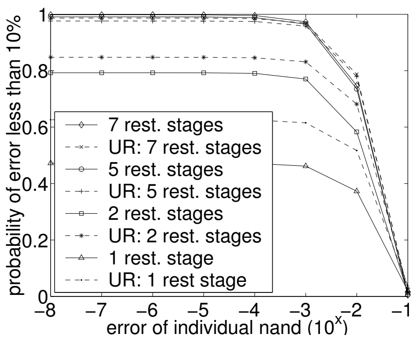
We have also modeled the injection of signal noise at the interconnects joining the different multiplexing stages, as well as at the primary inputs of the majority multiplexing configuration (wire error model). Signal noise may be characterized as follows– being a positive spike n_+, a negative spike n_- or causing no effect n_0. To illustrate our noise modeling methodology, let us walk thorugh an example. Assume x to be a signal at the output of one multiplexing stage. The signal goes through an interconnect (to the next stage) where there is noise n and x' is the signal at the other end of the interconnect. Suppose the probability of x being logic high is p and that of being logic low is q ($p + q = 1$). Due to the characterization of noise, signal x can either go from high to low, or low to high or remain the same due to the effect of n. Let the probabilities of n_+, n_- and n_0 be $p0$, $p1$ and $p2$ respectively ($p0 + p1 + p2 = 1$). Thus, the probability of x' being logic low is : $p(x' = 0) = p(x = 0)\cdot(n = n_0) + p(x = 0)\cdot(n = n_-) + p(x = 1)\cdot p(n_-) = q\cdot p2 + q\cdot p1 + p\cdot p1 = q\cdot p2 + p1\cdot(q+p) = q\cdot p2 + p1$. Similarly, the probability of x' being logic high can be computed. If $p0 = 0.05$, $p1 = 0.05$ and $p2 = 0.90$, then $p(x' = 0) = 0.9\cdot q + 0.05$. Our framework is highly parameterized, and the characteristics of such signal noise can be modified easily so as to observe the effect on the reliability measures of different multiplexing configurations. Similar to the previous experimental setups, we assume that two of the inputs to the majority gates have high probability of being logic high (0.9) while the third input has a 0.9 probability of being a logic low. The signal at the output of a gate is inverted if it is in error (von Neumann fault). Also, the probability values $p0$, $p1$ and $p2$ to characterize the signal noise in the experiments are similar to the ones assumed in the illustration above. This is a simple probabilistic noise model and more complex models can be implemented in our framework.

In Figure 8, the probabilities that less than 10% of the outputs are incorrect are plotted against small gate failure probabilities for multiplexing with 3, 4, 5 and 7 stages. It can be seen that for a particular bundle size, increasing the number of multiplexing stages yields better reliability (probability values increase). Also, variation of the gate failure does not significantly change the level of reliability at a particular redundancy level (i.e., restorative stage). However, when Figures 8 (a) and (b) are compared, it can be observed that the reliability decreases when the bundle size is increased (from 10 to 20) while keeping similar restorative stage and gate failure probability configurations. This result conforms with [5], where Pippenger showed that computation in the presence of noise (transient faults) requires more layers of redundancy. Similar results are observed for large gate failure probabilities, and due to space constrains, only plots for I/O bundle size of 10 is shown in Figure 7 (b).

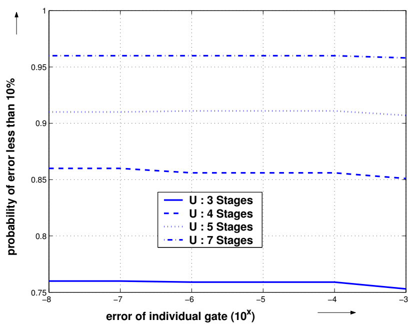
V. Comparison of Reliability Measures for NAND and Majority Multiplexing Systems

In this section we compare reliability measures of the multiplexing architectures for the NAND and majority gates. The input distributions for the majority multiplexing system (shown in Figure 2) are identical to the ones assumed in Section IV i.e., two of the inputs have high probability of being logic high (0.9) while the third input has a 0.9 probability of being a logic low. Thus the normal operation of the system should return a stimulated output. The reliability results for the NAND multiplexing configuration (Figure 1 (b)) reported in [7] assume that the inputs X and Y are identical and that initially 90% of the inputs are stimulated. It is also assumed that for these multiplexing configurations, the gate failure is a von Neumann fault (inverted output error).

Figure 9 compares the reliability of the majority and NAND multiplexing systems in terms of probability of

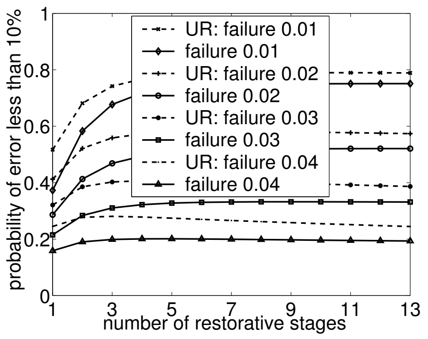


(a) NAND multiplexing [7]

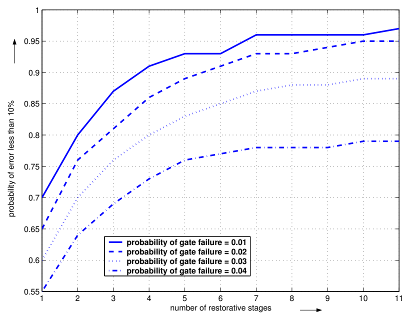


(b) Majority multiplexing

Fig. 9. Probability of at most 10% of the outputs being incorrect for the multiplexing systems when I/O bundle size equals 20 and gate failure probabilities are small



(a) NAND multiplexing [7]



(b) Majority multiplexing

Fig. 10. Probability of at most 10% of the outputs being incorrect for the multiplexing systems when I/O bundle size equals 20 and gate failure probabilities are large

at most 10% of the outputs being incorrect, for I/O bundle size of 20 and small gate failure probabilities. The number of restorative stages varies between 3 and 7. There are two sets of plots for each redundancy level (in terms of number of restorative stages) in Figure 9 (a). We only consider the plots for random selection of the input signals without replacement (UR that represents random choice with replacement is not considered). Comparing these results, we can infer that for both the multiplexing systems, increasing the number of stages enhances the reliability measures and that the rate of increase in reliability decreases as more restorative stages are added to the configurations. It can also be observed that NAND multiplexing provides better reliability than its majority gate counterpart, and such an observation is independent of the number of restorative stages.

Also, Figure 10 gives a comparison of the majority and NAND multiplexing systems in terms of reliability, for I/O bundle size of 20 and when probability of gate failure is large. The gate failure probabilities vary between 0.01 and 0.04. In Figure 10 (a), the plots for UR are ignored since our majority multiplexing model only performs random choice without replacement. Figure 10 (a) and (b) show that for both the multiplexing systems as the probability of gate failure increases, high reliability of computation cannot be achieved even by adding more restorative stages. This is because the reliability of the system reaches a saturation point at an optimal redundancy factor (number of restorative stages). The results reported also indicate that the majority multiplexing scheme provides better reliability than NAND multiplexing, and such an observation is independent of the number of restorative stages.

VI. Conclusion

This paper addresses the need to have automation methodologies for analyzing reliability of fault-tolerant architectural configurations. In view of the need to develop such automation, we have extended our reliability evaluation tool NANOPRISM [8] by developing a DTMC-based generic multiplexing framework. Either an entire Boolean network or a fragment of it can be plugged into such a framework to evaluate different redundancy and reliability trade-offs. As an illustration, a majority circuit is configured as a multiplexing system and different reliability measures are computed in the presence of different fault models (permanent and transient faults). When comparing these reliability-redundancy plots with the results reported on NAND multiplexing [7], we have determined a very interesting result: majority multiplexing provides better reliability for large gate failure probabilities, whereas when probability of gate failure is small, NAND multiplexing seems to perform better. Given the importance of majority gates in newer technologies, this is an important observation to keep in mind. Such an observation also indicates the effectiveness of the NANOPRISM multiplexing library.

References

[1] C. S. Lent, P. D. Tougaw, W. Porod, and G. H. Bernstein, "Quantum cellular automata," *Nanotechnology*, vol. 4, pp. 49–57, January 1993.

[2] C. Lent, "A device architecture for computing with quantum dots," in *Porceedings of the IEEE*, vol. 541, April 1997, p. 85.

[3] G. L. Snider, A. O. Orlov, I. Amlani, G. H. B. adn Craig S. Lent, J. L. Merz, and W. Porod, "Quatum-dot cellular automata: Line and majority logic gate," *Japanese Journal of Applied Physics*, vol. 38, no. 12B, pp. 7227–7229, December 1999.

[4] J. von Neumann, "Probabilistic logics and synthesis of reliable organisms from unreliable components," *Automata Studies*, pp. 43–98, 1956.

[5] N. Pippenger, "Reliable computation by formulas in the presence of noise," *IEEE Transactions on Information Theory*, vol. 34, no. 2, pp. 194–197, 1988.

[6] R. Dobrushin and E.Ortyukov, "Upper bound on the redundancy of self-correcting arrangements of unreliable functional elements," *Problems of Information Transmission*, vol. 13, no. 3, pp. 203–218, 1977.

[7] Omitted for Blind Review.

[8] Omitted for Blind Review.

[9] J. Han and P. Jonker, "A system architecture solution for unreliable nanoelectronic devices," *IEEE Transactions on Nanotechnology*, vol. 1, pp. 201–208, 2002.

[10] K. Nikolic, A. Sadek, and M. Forshaw, "Architectures for reliable computing with unreliable nanodevices," in *Proc. IEEE-NANO'01*. IEEE, 2001, pp. 254–259.

[11] W. Evans and N. Pippenger, "On the maximum tolerable noise for reliable computation by formulas," *IEEE Transactions on Information Theory*, vol. 44, no. 3, pp. 1299–1305, 1998.

[12] S. Roy and V. Beiu, "Majority multiplexing: Economical redundant fault-tolerant designs for nano architectures," *IEEE Transactions on Nanotechnology*, sent for Publication.

[13] Web Page: www.cs.bham.ac.uk/~dxp/prism/.

[14] M. Kwiatkowska, G. Norman, and D. Parker, "Prism: Probabilistic symbolic model checker," in *TOOLS 2002*, ser. LNCS, vol. 2324. Springer-Verlag, April 2002, pp. 200–204.

[15] W. Roscoe, *Theory and Practice of Concurrency*. Prentice Hall, 1998.

[16] H. Hansson and B. Jonsson, "A logic for reasoning about time and probability," *Formal Aspects of Computing*, vol. 6, no. 5, pp. 512–535, 1994.